（续）

方法	功能
lst. back( )	返回链表中最后一个元素的引用
lst. clear( )	擦除所有元素
lst. insert(pos, value)	在 pos 迭代器之前插入一个值为 value 的元素
lst. erase(pos)	从容器中删除位于 pos 的元素
lst. erase(first， last)	从容器中删除[first, last) 范围中的元素
lst. push_back(value)	把给定元素添加到容器尾
lst. push_front(value)	把给定元素添加到容器首
lst. pop_back( )	删除容器的首元素
lst. pop_front( )	删除容器的末元素
lst. resize(count)	重设容器大小以容纳 count 个元素。若当前大小大于 count，则减小容器为其首 count 个元素。若当前大小小于 count，则后附额外的默认插入的元素

4. 向量(vector)

向量是一个封装了动态大小数组的顺序容器(sequence container)。跟任意其他类型容器一样，它能够存放各种类型的对象。可以简单地认为，向量是一个能够存放任意类型的动态数组。容器定义在头文件 vector 中。对于 vector 类型的变量 vec，基本操作有如表 1.9 所示的几种。

表 1.9　向量的基本操作

方法	功能
vec. empty( )	判断是否为空
vec. push_back(x)	在尾部增加一个元素 x
vec. insert(it, x)	迭代器指向元素前增加一个元素 x
vec. insert(it, n, x)	迭代器指向元素前增加 n 个相同的元素 x
vec. insert(it, first, last)	迭代器指向元素前插入另一个相同类型向量的[first, last) 间的数据
vec. size( )	返回元素的个数
vec. max_size( )	返回最大可允许的元素数量值
vec. at(pos)	返回 pos 位置元素的引用
vec. front( )	返回首元素的引用
vec. back( )	返回尾元素的引用
vec. begin( )	返回头指针，指向第一个元素
vec. end( )	返回尾指针，指向最后一个元素的下一个位置
vec. rbegin( )	反向迭代器，指向最后一个元素
vec. rend( )	反向迭代器，指向第一个元素之前的位置
vec. erase(it)	删除迭代器指向元素
vec. erase(first, last)	删除[first, last) 中元素
vec. pop_back( )	删除最后一个元素
vec. clear( )	清空所有元素

参考词条

1. 线性结构
2. 双端队列(deque)、优先队列(priority_queue)

延伸阅读

[1] BJARNE S. C++程序设计语言(第1~3部分)[M]. 王刚，杨巨峰，译. 4版. 北京：机械工业出版社，2016：81-84.

[2] BJARNE S. C++程序设计语言(第4部分：标准库)[M]. 王刚，杨巨峰，译. 4版. 北京：机械工业出版社，2016：54-55.

(金靖)

1.3 数据结构

1.3.1 线性结构

1.3.1.1 链表：单链表、双向链表、循环链表

链表是一种在物理存储空间上不连续的存储结构，链表内元素的逻辑顺序是通过链表中的指针链接次序实现的。根据链表的指针链接方式，链表一般分为单链表、双向链表、循环链表三种形式。

1. 单链表

单链表也称为单向链表，其结点中包含数据域 data 与指针域 next，数据域用来存储数据，指针域用来链接下一个结点，如图 1.1 所示。

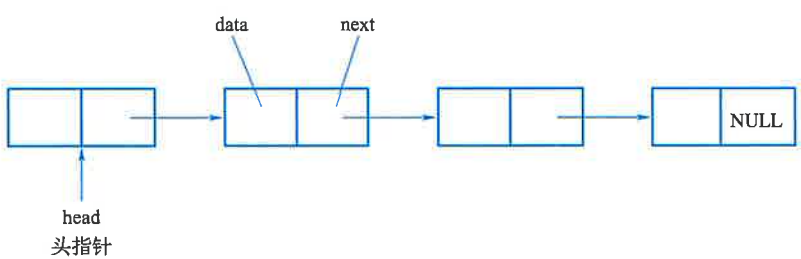

图 1.1 单链表

单链表一般有查找、插入、删除等操作，其中插入操作有头插法、尾插法两种方法。

2. 双向链表

双向链表结点中包含数据域 data 与两个指针域 prev 和 next，prev 指向当前元素的前驱元素，next 指向当前元素的后继元素，如果某个元素没有前驱元素，则该元素是链表的第一个元素，也就是头指针(head)，如果某个元素没有后继元素，则该元素是链表的最后一个元素，也就是尾指针(tail)。双向链表如图 1.2 所示。

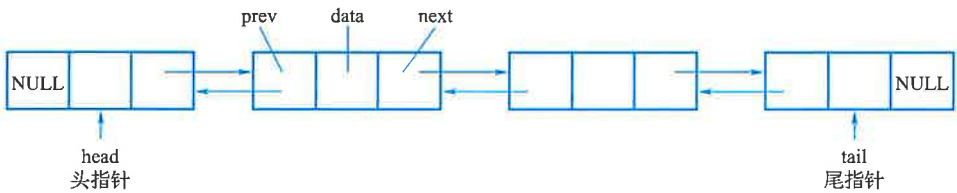

图 1.2　双向链表

双向链表一般有查找、插入、删除等操作。

3. 循环链表

循环链表分为单向循环链表和双向循环链表。

如果将单向链表中最后一个元素的指针指向第一个元素，则这样的单向链表就是单向循环链表。

如果将双向链表中第一个元素的前驱指针指向双向链表的最后一个元素，再将双向链表中最后一个元素的后继指针指向双向链表的第一个元素，那么这样的双向链表就是双向循环链表。

循环链表一般有查找、插入、删除等操作。

代码示例

使用结构体实现双向链表的数据结构，代码如下。

```
struct List
{
    int data,prev,next;
    List(){};
    List(int x,int a,int b) : data(x),prev(a),next(b){};
} t[1000005];
int tot;
```

在双向链表的结点 p 后面插入值为 x 的新结点，代码如下。

```
int np = ++tot;
int next = t[p]. next;
t[p]. next = np;
t[np] = List(x,p,next);
t[next]. prev = np;
```

删除双向链表中的结点 p，代码如下。

```
int prev = t[p]. prev,next = t[p]. next;
t[prev]. next = next;
t[next]. prev = prev;
```

参考词条

1. 数组与数组下标
2. 指针

延伸阅读

[1] THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：131-134.

[2] 严蔚敏，吴伟民. 数据结构[M]. 北京：清华大学出版社，2007：27-37.

[3] DONALD E K. 计算机程序设计艺术 卷 1：基本算法[M]. 李伯民，范明，蒋爱军，译. 3 版. 北京：人民邮电出版社，2016：217-236.

典型题目

CSP2021-J 小熊的果篮

（叶金毅）

1.3.1.2 栈

栈是一种限定仅在表尾进行插入和删除操作的线性表。一般情况下，操作端称为栈顶，另一端称为栈底。栈实现了一种后进先出(last-in，first-out，LIFO)的策略，栈的模型如图 1.3 所示。

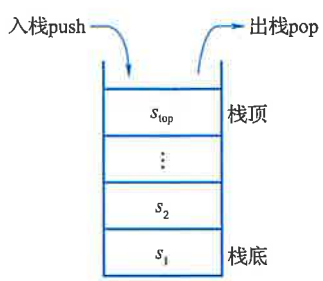

图 1.3 栈的模型

栈的基本操作包括入栈、出栈、取栈顶元素等。

代码示例

利用数组可以实现栈的数据结构及部分基本操作，代码如下。

```
const int maxsize = 1010;
int s[maxsize];
int top = 0;
//入栈操作
s[++top] = x;
//出栈操作
top--;
//取栈顶元素
int x = s[top];
```

参考词条

1. 数组与数组下标
2. 链表：单链表、双向链表、循环链表
3. 双端栈

延伸阅读

[1] THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：129-130.
[2] 严蔚敏，吴伟民. 数据结构[M]. 北京：清华大学出版社，2007：44-58.
[3] DONALD E K. 计算机程序设计艺术 卷 1：基本算法[M]. 李伯民，范明，蒋爱军，译. 3 版. 北京：人民邮电出版社，2016：191-194.

典型题目

1. NOIP2003 普及组 栈
2. NOIP2013 普及组 表达式求值

（叶金毅）

1.3.1.3 队列

队列是一种限定仅在队尾进行插入和在队头进行删除操作的线性表，队列实现了先进先出(first-in first-out，FIFO)策略，队列的模型如图 1.4 所示。

图 1.4 队列的模型

队列的基本操作包括入队、出队、取队首元素、取队尾元素等。

代码示例

利用数组可以实现队列的数据结构及部分基本操作，代码如下。

```
const int maxsize = 1010;
int q[maxsize];
int front = 1;
int tail = 0;
int x;
//入队列
q[++tail] = x;
//出队列
front++;
```